Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 1998 Proceedings

Americas Conference on Information Systems (AMCIS)

December 1998

Software Reuse Management: Development of a Model in the Context of the Capability Maturity Model

Anand Vadapalli
Cincinnati Bell Telephone

Derek Nazareth
University of Wisconsin-Milwaukee

Software Reuse Management: Development of a Model in the Context of the Capability Maturity Model

Anand Vadapalli
Cincinnati Bell Telephone

Derek L. Nazareth
University of Wisconsin-Milwaukee

Abstract

This paper describes a model for software reuse management within the context of the Capability Maturity Model. It outlines reuse components and practices as organizations move through more mature levels of software development. Unlike prior models that focus on a prescriptive approach, the reuse management model acknowledges the existence of a sizeable portfolio of existing applications, including legacy systems, which can provide basis for reuse.

Introduction

In the software development domain, software reuse is increasingly stated to be one of the fundamental paradigms of development, based on the belief that "the defining characteristic of good reuse is not the reuse of software per se, but the reuse of human problem solving" [Barnes & Bollinger 1991]. It has been estimated that only 15% of code in software application represents unique code [Tracz 1988]. Reinvention rather than reuse has been the norm in software development. With software reuse being a critical aspect of any development process, it follows that reuse management will be an important aspect of software process management. While it is currently popular to posit maturity models for various aspects of software development (including people [Curtis et. al. 1995], software acquisition [Ferguson et. al. 1996], individual skills [Prasad 1997], to name a few), we believe that placing reuse management within the context of an established framework like the CMM will be far more effective. The various types of reuse, their correspondence with the levels in CMM, and the management and measurement issues at each of the reuse stages will be highlighted.

Software Reuse

The motivating factors for increasing interest in software reuse are traditionally grounded in economic considerations – lower development costs, higher productivity, shorter development periods, reclamation of legacy system value, etc [Kasperson 1994]. Reuse also has the potential to improve software quality, reduce maintenance, reduce software risk, and promote interoperability, among others [Poulin 1997]. While the potential benefits from software reuse are undeniable, realizing these benefits pose some challenge. Documenting the extent of these benefits also remains a concern.

Examples of potentially reusable products include requirement specifications, designs, code modules, documentation, test data, and customized tools [Barnes & Bollinger 1991]. Several taxonomies of reuse are available which identify facets of reuse by substance, scope, mode, technique, intention and product [Prieto-Diaz 1993], [Frakes & Terry 1996].

While reuse at a coding or implementation level is prevalent in some form or the other in most development exercises, planned reuse at an enterprise level appears less widespread. Over the past few years, organization reports of reuse have been encouraging. Establishing the growing trend of software reuse only serves to reinforce the need to be able to manage the process of reuse.

Reuse Management Model

Prior models for software reuse have generally sought to describe the evolution of reuse in an organization, including a reuse maturity model [Kolton & Hudson 1991], and a reuse capability model [Davis 1993]. Despite their derivation from CMM principles, neither seeks to link the evolution of reuse to the CMM. We believe that a reuse management model should utilize a sound framework for its effective adoption. The model identifies the stages of software reuse, their correspondence to the CMM levels, and the management and measurement issues at each stage. Management commitment, by way of sustained support and investments is the first step to reuse, and is assumed to be a prerequisite to the model. The three different levels of reuse in the model are elaborated below.

Stage 1: Ad-hoc or Opportunistic Reuse

In this stage, reuse is typically an individual effort by developers, rather than an enterprise-wide phenomena. Typically, this involves reuse of code fragments, routines, procedures, and the like. Reuse tends to be in the *Product* category, with some

degree of reuse at the *Personnel* level, in terms of using the same personnel for similar software development, based on available skills and training.

This level of reuse is largely shaped by the individual programmer to recall his or her prior work, and use appropriate segments in the task at hand. Reuse components will likely include code fragments, ranging from function prototypes, file declaration structures, to entire modules or programs. Reuse occurs either through modification of existing software assets (more common) or through the development of assets with reuse in mind (less frequently). The emphasis at this level of reuse is the central role of the individual programmer rather than a planned or concerted effort by the organization.

Stage 2: Coordinated or Compositional Reuse

At the next level of the hierarchy, reuse is based on the existence of a library of routines and code available within the enterprise, which can be used by various developers. Additions to this library are not on a planned basis, but are the by-product of various development projects. In addition to *Product* and *Personnel* reuse, the existence of a library implies a set of standards for software development, including standardization at the *Process* level.

The management of a library of reusable software modules does raise some issues, including component naming conventions, policies for component inclusion, search and retrieval mechanisms, and modification criteria. The need to address interoperability issues, as well as version control, will also determine the effectiveness of successful software reuse at this level. The focus on reuse now shifts from an individual to a formal group within the enterprise.

Stage 3: Planned or Generative Reuse

As the highest level of reuse, system components are designed to be generic and widely applicable within their domain of applications. Concepts of domain engineering and analysis become critical. At this stage, *Product* reuse widens to include requirements and design specification reuse – providing the greatest leverage for reuse. *Process* and *Personnel* reuse, combined with the higher levels of *Product* reuse, contribute to major benefits achieved at this stage.

Relationship with CMM

Grounding the model within the context of the CMM provides a framework for assessment of its effectiveness. However, some notion of measuring and demonstrating the effectiveness of reuse is necessary for this. There are several models for measuring software reuse – excellent summaries are available in [Frakes & Terry 1996], [Poulin 1997]. These address some of the problematic issues as to what to measure, how to count, etc. For our purposes, we will simply classify the chosen metrics based upon their purpose – monitoring or predictive. Monitoring metrics address the following questions: What is the amount of reuse associated with a component? What is the quality of reuse? What are the time and cost savings that accrue from reuse? Predictive metrics on the other hand address the following questions: Can we predict the reusability of a component (and hence its value? How can we predict development costs and times if reuse is present?

In terms of mapping to the CMM, the reuse management model maps relatively cleanly, with ad-hoc reuse occurring in the *Initial* level, coordinated reuse occurring at the *Repeatable* and *Defined* levels, and planned reuse at the *Managed* and *Optimized* levels, as illustrated in Figure 1.

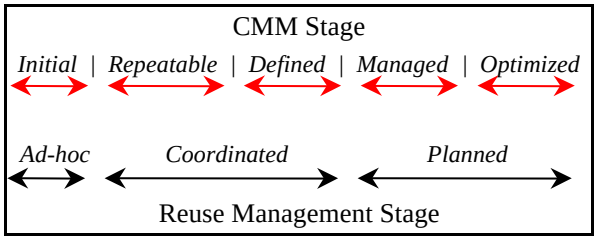

Figure 1. Reuse Management and CMM

At the *Initial* level, without a planning or a control system in place, an organization can hardly be expected to focus on reuse as a conscious development philosophy. However, a lot of software development involves the reuse of code fragments by individual programmers. Thus, the ad-hoc stage of reuse maps to this level of CMM. While there is potentially both personnel and product reuse, these efforts are more unplanned than systematic; and the organization is unlikely to accrue many benefits of reuse at this stage.

As an enterprise moves to the *Repeatable* and *Defined* levels of the CMM, a basic definition of the software process emerges, and the organization tends to reuse its process in addition to personnel and product. Personnel and product reuse will be more organized and are amenable to monitoring. The establishment of a Process Group at the *Defined* level will considerably aid in the collection and monitoring of reuse metrics, and the establishment of a component library. Library management may not yet be fully organized tasks, though this will certainly have an impact on effectiveness. This level still would largely involve code level product reuse. Measurement of the extent of reuse will likely dominate any reuse metrics adopted.

At the *Managed* and *Optimized* levels of the CMM, it is expected that from the earliest development stages, developers plan for reuse. The use of domain analysis, coupled with formal methods, and leveraging existing assets characterizes this stage. Automated tool usage in *Managed* and *Optimized* levels for recording various attributes of the software process will draw greater attention to reuse potential. Reuse metrics will now start to focus on the impact of reuse on the development process, expanding to include predictive metrics.

Table 1 summarizes this discussion.

Table 1. Reuse Management Model

Reuse Level	CMM Level	Reuse Asset	Key Practices	Reuse Metrics
Ad-hoc	• Initial	•Personnel •Product ♦code	•Recall-based reuse •Individual expertise •Code reuse	None
Coordinated	•Repeatable •Defined	• Process •Personnel •Product ♦code	•Library maintenance ♦definition ♦cataloging ♦retrieval • Interoperability standards •Version control	•Monitoring
Planned	•Managed •Optimized	•Process •Personnel •Product ♦requirements ♦design ♦code	• Domain analysis •Object definition •Formal methods •Interface with legacy sys. •Reverse engineering	•Monitoring •Predictive

Some Recurring Themes

Standards form an integral part of effective software reuse. While it is unlikely that any formal standards will be present at the ad-hoc level, some minimal standards are necessary at the coordinated level, and a formal mechanism for standards approval and enforcement is necessary for planned reuse. The adoption of universal standards will also play a significant role at this level.

While much of the CMM is directed at new software development, most organizations have a sizeable investment in existing systems that may not be immediately amenable to these new practices. The ability to reverse engineer legacy systems, and move them into the new practices is critical for effective adoption of the model.

Conclusions

The paper integrates existing knowledge on software process management and software reuse, and develops a macro-level reuse management model. The development of this model is an important contribution in terms of integrating existing research as well as providing practitioners a holistic view of the software reuse process.

References

References are available from the authors (avadapalli@cinbell.com; derek@uwm.edu).